X Congresso Brasileiro de Inteligência Computacional (CBIC'2011), 8 a 11 de Novembro de 2011, Fortaleza, Ceará


# APPLYING THE INCREMENTAL GAUSSIAN NEURAL NETWORK TO CONCEPT FORMATION AND ROBOTIC TASKS

**Milton Roberto Heinen, Paulo Martins Engel and Rafael C. Pinto**
Informatics Institute, Universidade Federal do Rio Grande do Sul (UFRGS)
{miheinen,engel,rcpinto}@inf.ufrgs.br

**Abstract –** IGMN (standing for Incremental Gaussian Mixture Network) is a new connectionist approach for incremental function approximation, concept formation and robotic tasks. It is based on strong statistical principles (Gaussian mixture models) and asymptotically converges to the optimal regression surface as more training data arrive. Moreover, IGMN learns instantaneously and incrementally using a single scan over the training data, it can handle the stability-plasticity dilemma and does not suffer from catastrophic interference. The main goal of this paper is to demonstrate the suitability of IGMN in on-line and critical control applications such as incremental concept formation and robotic tasks. Through several experiments, described in this paper, it is shown that IGMN is a very useful machine learning tool for this kind of tasks. Moreover, it does not require fine-tuning its configuration parameters, it is not sensitive to initialization conditions and has a good computational performance, thus allowing its use in real time control applications.
**Keywords –** Artificial neural networks, Bayesian methods, concept formation, incremental learning, Gaussian mixture models, autonomous robots.

**Resumo –** IGMN (do inglês *Incremental Gaussian Mixture Network*) é uma nova abordagem conexionista para a aproximação de incremental de funções, formação de agrupamentos e robótica. Ele é baseado em fortes princípios estatísticos (modelos de mistura gaussianos) e assintoticamente converge para a superfície de regressão ótima a medida que os dados de treinamento chegam. Além disso, o IGMN aprende de forma incremental utilizando uma única passada sobre os dados de treinamento, resolve o dilema plasticidade estabilidade e não sofre de interferência catastrófica. O principal objetivo deste artigo é demonstrar o uso do IGMN em aplicações de robótica e controle que exigem aprendizado incremental e em tempo real, tais como a formação incremental de conceitos e robótica móvel. Através de diversos experimentos descritos neste artigo é demonstrado que o IGMN é uma ferramenta de aprendizado de máquina bastante útil neste tipo de tarefa. Além disso, ele não requer ajustes finos em seus parâmetros de configuração, não é sensível as condições de inicialização e possui um bom desempenho computacional, o que permite o seu uso em aplicações de controle e em tempo real.
**Palavras-chave –** Redes neurais artificiais, métodos Bayesianos, formação de conceitos, aprendizado incremental, modelos de mistura gaussianos, robôs autônomos.

## 1 Introduction

Artificial neural networks (ANNs) [1] are mathematical or computational models inspired by the structure and functional aspects of biological neural networks. They are composed by several layers of massively interconnected processing units, called artificial neurons, which can change their connection strength (i.e., the synaptic weights values) based on external or internal information that flows through the network during learning. Although artificial neural networks (ANNs) have been successfully used in several tasks, including signal processing, pattern recognition and robotics, most ANN models have some disadvantages that difficult their use in on-line tasks such as incremental concept formation and real-time robotic control. The Backpropagation learning algorithm, for instance, requires several scans over all training data, which must be complete and available at the beginning of the learning process, to converge for a good solution. Moreover, after the end of the training process the synaptic weights are "frozen", i.e., the network loses its learning capabilities.

These drawbacks highly contrast with the human brain learning capabilities because: (i) we don't need to perform thousands of scans over the training data to learn something (in general we are able to learn using few examples and/or repetitions); (ii) we are always learning new concepts as new "training data" arrive, i.e., we are always improving our performance through experience; and (iii) we don't have to wait until sufficient information arrives to make a decision, i.e., we can use partial information as it becomes available. Besides being not biologically plausible, these drawbacks difficult the use of ANNs in on-line robotics because in this kind of application the training data are just instantaneously available to the learning system, and in general a decision must be made using the information available at the moment.

In [2, 3] a new artificial neural network model, called IGMN (standing for Incremental Gaussian Mixture Network), is proposed to tackle great part of these problems described above. IGMN is based on parametric probabilistic models (Gaussian mixture models), that have nice features from the representational point of view, describing noisy environments in a very parsimonious way, with parameters that are readily understandable. Moreover, IGMN has some useful features that are not present in other neural network models such as: (i) IGMN learns incrementally using a single scan over the training data; (ii) the learning process can proceed perpetually as new training data arrive; (iii) it can handle the stability-plasticity dilemma and does not suffer

from catastrophic interference; (iv) the neural network topology is defined automatically and incrementally; and (v) IGMN is not sensitive to initialization conditions. In [3, 4] IGMN was used in function approximation tasks using just synthetic data. This paper describes the use of IGMN in more realistic tasks such as incremental concept formation and robotics. Moreover, the obtained results are compared with other ANN models, thus pointing out the main advantages and suitability of IGMN. The remaining of this paper is organized as follows. Section 2 presents the IGMN architecture and its learning algorithm; Section 3 describes some experiments performed to evaluate IGMN in incremental concept formation and robotic tasks; and Section 4 provides some final remarks and perspectives.

## 2 Incremental Gaussian Mixture Network

Figure 1 shows the general architecture of IGMN. It is composed by an association region $\mathcal{P}$ (in the top of this figure) and many cortical regions, $\mathcal{N}^{\mathcal{A}}, \mathcal{N}^{\mathcal{B}}, \ldots, \mathcal{N}^{\mathcal{S}}$. All regions have the same number of neurons, $M$. Initially there is a single neuron in each region (i.e., $M = 1$), but more neurons are incrementally added when necessary using an error driven mechanism. Each cortical region $\mathcal{N}^{\mathcal{K}}$ receives signals from the $k$th sensory/motor modality, $\mathbf{k}$ (in IGMN there is no difference between sensory and motor modalities), and hence there is a cortical region for each sensory/motor modality.

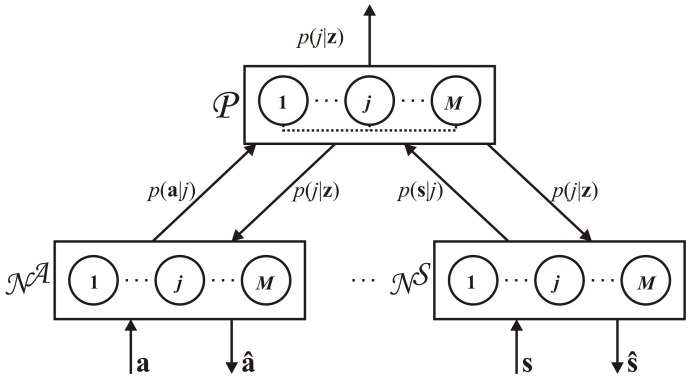



Figure 1: General architecture of IGMN

Another important feature of IGMN is that all cortical regions $\mathcal{N}$ execute a common function, i.e., they have the same kind of neurons and use the same learning algorithm. Moreover, all cortical regions can run in parallel, which improves the performance specially in parallel architectures. Each neuron $j$ of region $\mathcal{N}^{\mathcal{K}}$ performs the following operation:

$$p(\mathbf{k}|j) = \frac{1}{(2\pi)^{D^{\mathcal{K}}/2}\sqrt{|\mathbf{C}_j^{\mathcal{K}}|}} \exp\left\{-\frac{1}{2}(\mathbf{k}-\boldsymbol{\mu}_j^{\mathcal{K}})^T \mathbf{C}_j^{\mathcal{K}^{-1}}(\mathbf{k}-\boldsymbol{\mu}_j^{\mathcal{K}})\right\}, \tag{1}$$

i.e., a multivariate Gaussian distribution, where $D^{\mathcal{K}}$ is the dimensionality of $\mathbf{k}$ (different sensory/motor modalities $\mathbf{k}$ can have different dimensions $D^{\mathcal{K}}$). Each neuron $j$ maintains a mean vector $\boldsymbol{\mu}_j^{\mathcal{K}}$ and a covariance matrix $\mathbf{C}_j^{\mathcal{K}}$.

In IGMN the regions are not fully connected, i.e., the neuron $j$ of $\mathcal{N}^{\mathcal{K}}$ is connected just to the $j$th neuron of $\mathcal{P}$, but this connection is bidirectional. It is important to notice that there are no *synaptic weights* in these connections, i.e., all IGMN parameters are stored in the neurons themselves. A bottom-up connection between $\mathcal{N}^{\mathcal{K}}$ and $\mathcal{P}$ provides the component density function $p(\mathbf{k}|j)$ to the $j$th neuron in $\mathcal{P}$. Therefore, a neuron $j$ in the association region $\mathcal{P}$ is connected with the $j$th neuron of all cortical regions $\mathcal{N}$ via bottom-up connections and computes the a posteriori probability using the Bayes' rule:

$$p(j|\mathbf{z}) = \frac{p(\mathbf{a}|j)\, p(\mathbf{b}|j) \ldots p(\mathbf{s}|j)\; p(j)}{\sum_{q=1}^{M} p(\mathbf{a}|q)\, p(\mathbf{b}|q) \ldots p(\mathbf{s}|q)\; p(q)}, \tag{2}$$

where it is considered that the neural network has an arbitrary number, $s$, of cortical regions and $\mathbf{z} = \{\mathbf{a}, \mathbf{b}, \ldots, \mathbf{s}\}$. The dotted lines in Figure 1 indicate the lateral interaction among the association units for computing the denominator in (2).

Each neuron $j$ of the association region $\mathcal{P}$ maintains its a priori probability, $p(j)$, an accumulator of the a posteriori probabilities, $sp_j$, and an association matrix to store the correlations among each sensory/motor modality. If a neural network has two cortical regions, $\mathcal{N}^{\mathcal{A}}$ and $\mathcal{N}^{\mathcal{B}}$, for instance, then the association matrix $\mathbf{C}_j^{\mathcal{AB}}$ will have two dimensions and size $D^{\mathcal{A}} \times D^{\mathcal{B}}$. Note that it is not necessary to maintain $\mathbf{C}_j^{\mathcal{BA}}$ because $\mathbf{C}_j^{\mathcal{BA}} = \mathbf{C}_j^{\mathcal{AB}^T}$. The top-down connections between $\mathcal{P}$ and $\mathcal{N}^{\mathcal{K}}$, on the other hand, returns *expectations* to $\mathcal{N}^{\mathcal{K}}$ which are used to estimate $\hat{\mathbf{k}}$ when $\mathbf{k}$ is missing. This architecture is inspired on the memory-prediction framework (MPF) [5], which states that different cortical regions are not fully connected in the neocortex. Instead, they are linked to the association areas $\mathcal{P}$ through bottom-up and top-down connections, thus providing predictions and expectations, respectively, to all cortical regions $\mathcal{N}^{K}$. The main advantage of this strategy is to speed up IGMN and make it more suitable to real-time and critical applications, because it is much faster to invert two covariance matrices of size $M$ than a single covariance matrix of size $2M$. Moreover, a large number of samples is required to obtain good estimates from a large covariance matrix, and therefore using this strategy IGMN becomes more aggressive, i.e., it is able to provide good estimates using few training samples. Next subsections describe the IGMN operation in details.

### 2.1 IGMN operation

IGMN has two operation modes, called *learning* and *recalling*. But unlike most ANN models, in IGMN these operations don't need to occur separately, i.e., the learning and recalling modes can be intercalated. In fact, even after the presentation of a single training pattern the neural network can already be used in the recalling mode (the acquired knowledge can be immediately used), and the estimates become more precise as more training data are presented. Moreover, the learning process can proceed perpetually, i.e., the neural network parameters can always be updated as new training data arrive.

The following subsections describe the IGMN operation during learning and recalling. To simplify our explanation, we will consider that the neural network has just two cortical regions, $\mathcal{N}^{\mathcal{A}}$ and $\mathcal{N}^{\mathcal{B}}$, that receive the stimuli $\mathbf{a}$ and $\mathbf{b}$, respectively. It will be also considered that we are estimating $\hat{\mathbf{b}}$ from $\mathbf{a}$ in the recalling mode. But it is important to remember that: (i) IGMN can have more than two cortical regions (one for each sensory/motor stimulus $\mathbf{k}$); and (ii) after training it can be used to estimate either $\hat{\mathbf{a}}$ or $\hat{\mathbf{b}}$ (i.e., there is no difference between *inputs* and *outputs* in IGMN).

### 2.2 Learning mode

The learning algorithm used by IGMN is based on IGMM, presented in [6,7], but it has many modifications which adapt it to supervised tasks such as incremental function approximation and prediction. Before learning starts the neural network is empty, i.e., all regions have $M = 0$ neurons. When the first training pattern $\mathbf{z}^1 = \{\mathbf{a}^1, \mathbf{b}^1\}$ arrives, a neuron in each region is created centered on $\mathbf{z}^1$ and the neural network parameters are initialized as follows:

$$M = 1; \quad sp_1 = 1.0; \quad p(1) = 1.0; \quad \mathbf{C}_1^{\mathcal{AB}} = \mathbf{0};$$
$$\boldsymbol{\mu}_1^{\mathcal{A}} = \mathbf{a}^1; \quad \boldsymbol{\mu}_1^{\mathcal{B}} = \mathbf{b}^1; \quad \mathbf{C}_1^{\mathcal{A}} = {\boldsymbol{\sigma}_{ini}^{\mathcal{A}}}^2\mathbf{I}; \quad \mathbf{C}_1^{\mathcal{B}} = {\boldsymbol{\sigma}_{ini}^{\mathcal{B}}}^2\mathbf{I},$$

where the subscript '1' indicates the neuron $j = 1$ in each region, $M$ is the number of neurons in each region (all regions have the same size $M$), $sp$ is an accumulator of posterior probabilities maintained in the association region $\mathcal{P}$, $\mathbf{0}$ is a zero matrix of size $D^{\mathcal{A}} \times D^{\mathcal{B}}$, $\boldsymbol{\sigma}_{ini}^{\mathcal{A}}$ and $\boldsymbol{\sigma}_{ini}^{\mathcal{B}}$ are diagonal matrices that define the initial radius of the covariance matrices (the pdf is initially circular but it changes to reflect the actual data distribution as new training data arrive) and $\mathbf{I}$ denotes the identity matrix. $\boldsymbol{\sigma}_{ini}^{\mathcal{A}}$ and $\boldsymbol{\sigma}_{ini}^{\mathcal{B}}$ are initialized using a user defined fraction $\delta$ of the overall variance (e.g., $\delta = 1/100$) of the corresponding attributes, estimated from the range of these values according to:

$$\boldsymbol{\sigma}_{ini}^{\mathcal{K}} = \delta \left[\max(\mathbf{k}) - \min(\mathbf{k})\right] , \tag{3}$$

where $[\min(\mathbf{k}), \max(\mathbf{k})]$ defines the domain of a sensory/motor modality $\mathbf{k}$ (throughout this paper the symbol $k$ will be used to indicate any sensory/motor modality, i.e., either $a$ or $b$ in this case). It is important to say that it is not necessary to know the *exact* minimum and maximum values along each dimension to compute $\boldsymbol{\sigma}_{ini}^{\mathcal{K}}$, but instead just the approximate domain of each feature.

When a new training pattern $\mathbf{z}^t$ arrives, all cortical regions are activated, i.e., $p(\mathbf{k}|j)$ is computed using Equation 1 above, and the probabilities $p(\mathbf{z}^t|j)$ are sent to the association region $\mathcal{P}$, which computes the *joint* posterior probabilities $p(j|\mathbf{z}^t)$ using the Bayes' rule in (2). After this, the posterior probabilities $p(j|\mathbf{z}^t)$ are sent back to all cortical regions, i.e., $\mathcal{N}^{\mathcal{A}}$ and $\mathcal{N}^{\mathcal{B}}$, which compute their estimates as follows:

$$\hat{\mathbf{b}} = \textstyle\sum_{j=1}^{M} p(j|\mathbf{z}^t)[\boldsymbol{\mu}_j^{\mathcal{B}} + \mathbf{C}_j^{\mathcal{BA}}{\mathbf{C}_j^{\mathcal{A}}}^{-1}(\mathbf{a}^t - \boldsymbol{\mu}_j^{\mathcal{A}})] \qquad \hat{\mathbf{a}} = \textstyle\sum_{j=1}^{M} p(j|\mathbf{z}^t)[\boldsymbol{\mu}_j^{\mathcal{A}} + \mathbf{C}_j^{\mathcal{AB}}{\mathbf{C}_j^{\mathcal{B}}}^{-1}(\mathbf{b}^t - \boldsymbol{\mu}_j^{\mathcal{B}})], \tag{4}$$

where $\mathbf{C}_j^{\mathcal{AB}}$ is the $j$th association matrix maintained in association region $\mathcal{P}$ and $\mathbf{C}_j^{\mathcal{BA}}$ is its transpose. Note that as the covariance matrices $\mathbf{C}_j^{\mathcal{A}}$ and $\mathbf{C}_j^{\mathcal{B}}$ were already inverted when $\mathcal{N}^{\mathcal{A}}$ and $\mathcal{N}^{\mathcal{B}}$ were activated in the bottom-up direction, we don't need to invert them again, i.e., we can temporarily store the corresponding inverse matrices to speed up the IGMN learning algorithm. Using the estimates $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}$ the normalized approximation error $\varepsilon$ is given by:

$$\varepsilon = \max_{\mathbf{k} \in \mathbf{z}} \left\{ \max_{i \in D^{\mathcal{K}}} \left[ \frac{\|k_i^t - \hat{k}_i\|}{\max(k_i) - \min(k_i)} \right] \right\} \tag{5}$$

where $[\min(k_i), \max(k_i)]$ defines the domain of the sensory/motor feature $k_i$. Again $\min(k_i)$ and $\max(k_i)$ do not need to be the exact minimum and maximum values of $\mathbf{k}$ – they may be just approximations of the domain of each $k_i$ feature (in fact $\min(k_i)$ and $\max(k_i)$ are used just to make IGMN more independent from the range of the data features). If $\varepsilon$ is larger than a user specified threshold, $\varepsilon_{max}$, than $\mathbf{z}^t$ is not considered as *represented* by any existing cortical neuron in $\mathcal{N}^{\mathcal{K}}$. In this case, a new unit $j$ is created in each region and centered on $\mathbf{z}^t$ and all priors of the association region $\mathcal{P}$ are recomputed by:

$$p(j)^* = \frac{sp_j}{\sum_{q=1}^{M^*} sp_q}. \tag{6}$$

Otherwise (if $\mathbf{z}$ is well explained by any of the existing Gaussian units), the a posteriori probabilities $p(j|\mathbf{z}^t)$ are added to the current value of the $sp(j)$ on the association region:

$$sp_j^* = sp_j + p(j|\mathbf{z}^t), \qquad \forall j, \tag{7}$$

and the priors $p(j)$ are recomputed using (6). Then $\omega_j = p(j|\mathbf{z}^t)/sp_j^*$ is sent back to all cortical regions, and the parameters of all neurons in $\mathcal{N}^{\mathcal{K}}$ are updated using the following recursive equations derived in [2, 6, 7]:

$$\boldsymbol{\mu}_j^{\mathcal{K}^*} = \boldsymbol{\mu}_j^{\mathcal{K}} + \omega_j \left(\mathbf{z}^t - \boldsymbol{\mu}_j^{\mathcal{K}}\right) \tag{8}$$

$$\mathbf{C}_j^{\mathcal{K}^*} = \mathbf{C}_j^{\mathcal{K}} - (\boldsymbol{\mu}_j^{\mathcal{K}^*} - \boldsymbol{\mu}_j^{\mathcal{K}})(\boldsymbol{\mu}_j^{\mathcal{K}^*} - \boldsymbol{\mu}_j^{\mathcal{K}})^T + \omega_j \left[(\mathbf{z} - \boldsymbol{\mu}_j^{\mathcal{K}^*})(\mathbf{z} - \boldsymbol{\mu}_j^{\mathcal{K}^*})^T - \mathbf{C}_j^{\mathcal{K}}\right], \tag{9}$$

where the superscript '$*$' refers to the new (updated) values. Finally the association matrix $\mathbf{C}_j^{\mathcal{AB}}$ is updated using the following recursive equation:

$$\mathbf{C}_j^{\mathcal{AB}^*} = \mathbf{C}_j^{\mathcal{AB}} - (\boldsymbol{\mu}_j^{\mathcal{A}^*} - \boldsymbol{\mu}_j^{\mathcal{A}})(\boldsymbol{\mu}_j^{\mathcal{B}^*} - \boldsymbol{\mu}_j^{\mathcal{B}})^T + \omega_j \left[(\mathbf{a}^t - \boldsymbol{\mu}_j^{\mathcal{A}^*})(\mathbf{b}^t - \boldsymbol{\mu}_j^{\mathcal{B}^*})^T - \mathbf{C}_j^{\mathcal{AB}}\right], \tag{10}$$

which is derived using the same principles described in [6, 7]. This equation allows computing the covariances among distinct cortical regions incrementally, and thus estimating a missing stimulus $\mathbf{k}$ without having to maintain and invert a complete variance/covariance matrix. In fact, using (10) the complete variance/covariance matrix is broken down in separate submatrices that can be efficiently maintained.

### 2.3 Recalling mode

In the recalling mode, a stimulus (e.g., $\mathbf{a}$) is presented to a *partially* trained neural network (as the learning process proceeds perpetually, in IGMN we never consider that the training process is over), which computes an estimate for another stimulus (e.g., $\hat{\mathbf{b}}$). As said before, IGMN can be used to estimate either $\hat{\mathbf{a}}$ or $\hat{\mathbf{b}}$, but to simplify our explanation in this and the following sections we will consider that we are estimating $\hat{\mathbf{b}}$ from $\mathbf{a}$.

Initially the stimulus $\mathbf{a}$ is received in the cortical region $\mathcal{N}^{\mathcal{A}}$, where each neuron $j$ computes $p(\mathbf{a}|j)$ using (1). These predictions are sent to the association region $\mathcal{P}$ through the bottom-up connections, which is activated using just $p(\mathbf{a}|j)$:

$$p(j|\mathbf{a}) = \frac{p(\mathbf{a}|j)\,p(j)}{\sum_{q=1}^{M} p(\mathbf{a}|q)\,p(q)}. \tag{11}$$

After this, $p(j|\mathbf{a})$ is sent to the cortical region $\mathcal{N}^{\mathcal{B}}$ via the top-down connections, and $\mathcal{N}^{\mathcal{B}}$ computes the estimated stimulus $\hat{\mathbf{b}}$ using the following equations [2]:

$$\overline{\mathbf{x}}_j^{\mathcal{B}} = \boldsymbol{\mu}_j^{\mathcal{B}} + \mathbf{C}_j^{\mathcal{B}\mathcal{A}}\mathbf{C}_j^{\mathcal{A}^{-1}}(\mathbf{a} - \boldsymbol{\mu}_j^{\mathcal{A}}) \qquad\qquad \hat{\mathbf{b}} = \textstyle\sum_{j=1}^{M} p(j|\mathbf{a})\,\overline{\mathbf{x}}_j^{\mathcal{B}}. \tag{12}$$

The main drawback of IGMN is that it is necessary to invert the covariance matrices $\mathbf{C}_j^{\mathcal{K}}$ for each training sample $(\mathbf{a}, \mathbf{b}, \ldots, \mathbf{k})$, and this requires $D^{\log_2 7}$ operations using the Strassen algorithm [8]. A way to reduce the computational complexity is to separate the data features in more sensory/motor areas, thus dividing a big covariance matrix in many matrices of smaller size.

## 3 Experiments

This section describes the use of IGMN in some applications such as concept formation and robotic control. The main goal of these experiments is to validate IGMN in real applications and/or using real data and also to demonstrate its suitability in many potential applications that require incremental learning and real time performance. This section is structured as follows. Subsection 3.1 discusses the use of IGMN for incremental concept formation, which is an important task in machine learning and robotics. Subsection 3.2 shows how IGMN can be used to compute the control actions for a mobile robot performing a wall following behavior.

### 3.1 Incremental concept formation

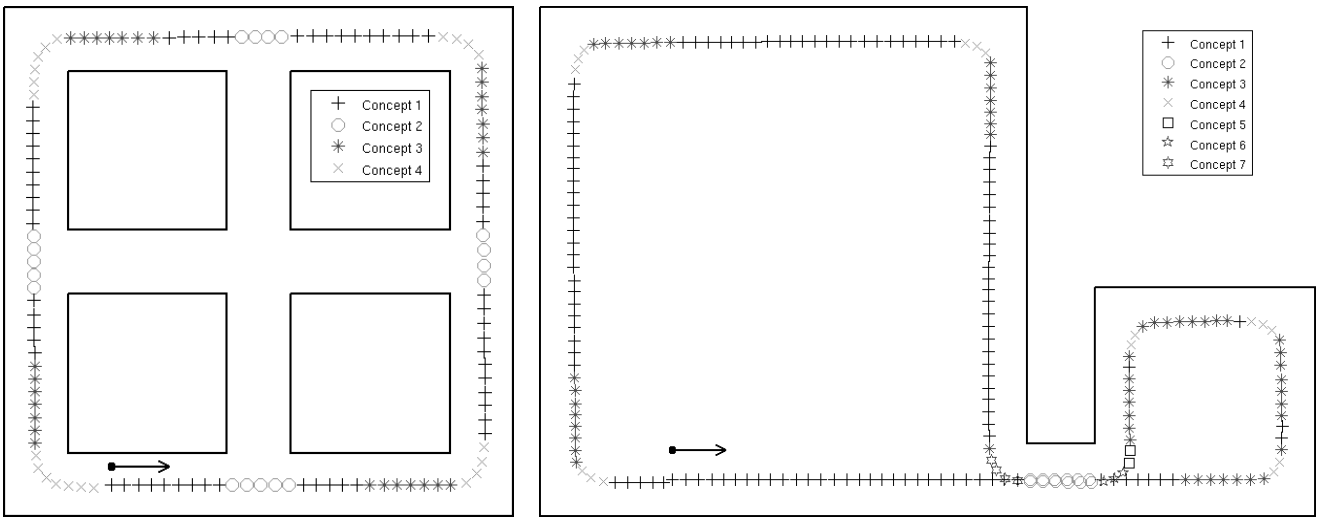
One of our primary motivations in developing IGMN was to tackle problems like those encountered in autonomous robotics. To be more specific, let us consider the so called perceptual learning, which allows an embodied agent to understand the world [9]. Here an important task is the detection of concepts such as "corners", "walls" and "corridors" from the sequence of noisy sensor readings (e.g., sonar data) of a mobile robot. The detection of these regularities in data flow allows the robot to localize itself and to detect changes in the environment [10].

Although concept formation has a long tradition in machine learning literature, in the field of unsupervised learning, most methods assume some restrictions in the probabilistic modeling [11] which prevent their use in on-line tasks. The Incremental Gaussian Mixture Network (IGMN), on the other hand, is able to learn from data flows in an incremental (new concepts can be added by demand) and on-line (it does not require that the complete training set be previously known and fixed) way, which makes it a good solution for concept formation in on-line robotic tasks. Moreover, unlike the traditional neural network models (e.g., MLP and GRNN), the IGMN hidden neurons are not "black boxes", and thus the Gaussian units can be interpreted as representations of the input space, i.e., high level concepts.

In the experiments described in this section the data consist of 10 continuous values provided by the Pioneer 3-DX simulator software ARCOS (Advanced Robot Control & Operations Software). The IGMN network used in these experiments has two

cortical regions, $\mathcal{N}^{\mathcal{S}}$ and $\mathcal{N}^{\mathcal{V}}$. The cortical region $\mathcal{N}^{\mathcal{S}}$ tackles the values of the sonar readings, i.e., $\mathbf{s} = \{s_1, s_2, \ldots, s_8\}$, and the cortical region $\mathcal{N}^{\mathcal{V}}$ receives the speeds applied at the robot wheels at time $t$, i.e., $\mathbf{v} = \{v_1, v_2\}$. To decide what is the most active concept at time $t$, the maximum likelihood (ML) hypothesis $\ell = \arg\max_j [p(j|\mathbf{z})]$, where $\mathbf{z} = \{\mathbf{s}, \mathbf{v}\}$, is used. It is important to note that IGMN computes and maintains the a posteriori probabilities of all concepts at each time, and hence it can be used in applications such as the so called multi-hypothesis tracking problem in robotic localization domains [10, 12]. The configuration parameters used in the following experiments are $\delta = 0.01$ and $\varepsilon_{max} = 0.1$. It is important to say that no exhaustive search was performed to optimize the configuration parameters.

The first experiment was accomplished in an environment composed of six corridors (four external and two internal), and the robot performed a complete cycle in the external corridors. Figure 2(a) shows the segmentation of the trajectory obtained by IGMN when the robot follows the corridors of this environment. IGMN created four units, corresponding to the concepts "corridor" (1: plus sign), "wall at right" (2: circle), "corridor / obstacle front" (3: asterisk) and "curve at left" (4: cross). The symbols in the trajectory of Figure 2(a) represent the ML hypothesis in each robot position, and the black arrow represents the robot starting position and direction.




(a) Front view (b) Array of sonar sensors

Figure 2: Segmentation obtained by IGMN

The next experiment was performed in a more complex environment, composed by two different sized rooms connected by a short corridor. This environment was originally used in [13] and [14, 15]. Figure 2(b) shows the segmentation performed by IGMN in this experiment. IGMN has created seven clusters, corresponding to the concepts "wall at right" (1: plus sign), "corridor" (2: circle), "wall at right / obstacle front" (3: asterisk), "curve at left" (4: cross), "bifurcation / obstacle front" (5: square), "bifurcation / curve at right" (6: five-pointed star) and "wall at left / curve at right" (7: hexagram).

Comparing these experiments, it can be noticed that some similar concepts, like "curve at left" and "obstacle front", were discovered in both experiments, although these environments are different (the environment shown in Figure 2(a) has many corridors whilst that one shown in Figure 2(b) has two large rooms and just one short corridor). This points out that concepts extracted from a data flow corresponding to a specific sensed environment are not restricted to this environment, but they form an alphabet that can be reused in other contexts. This is a useful aspect, that can improve the learning process in more complex environments.

As described above, the main goal in concept formation is to identify natural groupings in the input space, which are represented by IGMN using its Gaussian units. Therefore, in this kind of tasks the estimation/prediction capabilities of IGMN are not necessary. Nevertheless, these estimation/prediction capabilities can be used to compute the motor actions for a robot performing a wall following behavior, for instance, as is shown in the next subsection.

### 3.2 Estimating the desired speeds in a mobile robotics application

In the experiments described in the previous section the main goal was to identify natural groupings in the input space, which were represented by the IGMN Gaussian units, and therefore the IGMN estimation/prediction capabilities were not used. In this section we will use these estimation/prediction capabilities to compute the desired actions (i.e., the wheel speeds) for a mobile robot performing a wall following behavior in a simulated environment. These experiments are relevant because in robotic control tasks usually it is not possible to predict all situations that occur in the real world, and hence the robot needs to learn from experience while interacting with the environment.

In the next experiment IGMN was used to estimate the desired speeds in the same environment described above (Figure 2(b)) and using a IGMN neural network with two cortical regions, $\mathcal{N}^{\mathcal{S}}$ and $\mathcal{N}^{\mathcal{V}}$. Figure 3 shows the results obtained in this experiment, where the $x$ axis corresponds to the time index of the sensor readings (the training database is composed by 2070 data samples)

and the $y$ axis corresponds to the difference between the right and left motor speeds, i.e., $y_d(t) = v_1 - v_2$. A positive value in $y_d(t)$ corresponds to a left turn in the robot trajectory and a negative value corresponds to a right turn. The solid gray line in Figure 3 represents the desired $y_d(t)$ values and the dashed black line represents the difference between the actual IGMN outcomes, i.e.: $y_o(t) = \hat{v}_1$ - $\hat{v}_2$. It is important to say that the region $\mathcal{N}^{\mathcal{V}}$ has size $D^{\mathcal{V}} = 2$, i.e., the difference $y_o(t)$ was computed just to improve the visualization in Figure 3.

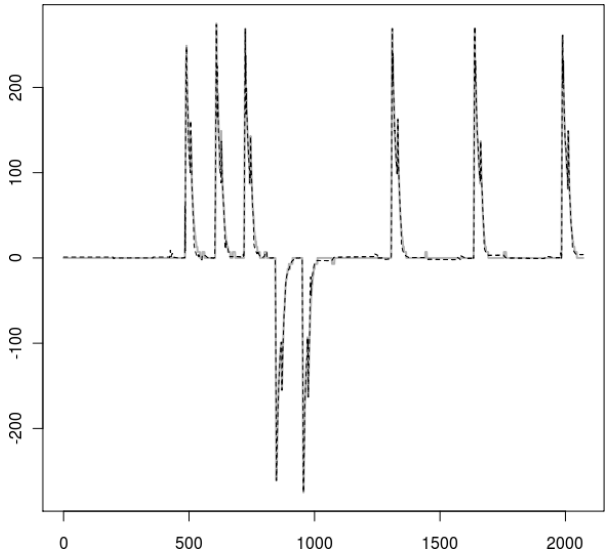



Figure 3: Difference between the speeds of the right and left motors while it is fallowing the trajectory shown in Figure 2(b).

The configuration parameters used in this experiment are $\delta = 0.01$ and $\varepsilon_{max} = 0.01$. We can notice in this figure that the approximation is quite good: the NRMS error is just 0.034598. The time required for learning was 0.351 seconds, and 40 Gaussian units were added in each region. Moreover, if we use this trained network to control the robot, it will follow the same "target trajectory" shown in Figure 2(b). These results are still more impressive because the robot does not receive any information about its position (just the readings of the sonar sensors are provided at each 100 milliseconds) and the neural network has no memory of past perceptions and actions, i.e., an action must be taken using just the current perception.

Table 1 shows a comparison among the results obtained using IGMN and other neural approaches. The first column presents the ANN model. The following columns show, respectively, the number of hidden/pattern units, the NRMS error computed using the 10-fold cross validation procedure, the average number of training epochs and the time required to perform each replication, i.e. 1/10 of the time required by the entire 10-fold cross validation procedure. To facilitate our comparison, the last row on Table 1 shows the results obtained using IGMN. Figure 4 shows a boxplot graph comparing the NRMS errors in these experiments. To allow a fair comparison, an exhaustive search was performed to find out the best configuration parameters of each ANN model.

Table 1: Comparative among ANNs in follow the trajectory of Figure 2(b)

| **ANN model** | **Units** | **NRMS** | **Epochs** | **Time** |
|---|---|---|---|---|
| MLP – RPROP | 10 | 0.043192 | 236.4 | 5.367s |
| MLP – LM | 10 | 0.034778 | 53.7 | 4.897s |
| GRNN ($\sigma = 90$) | 2070 | 0.033172 | 1.0 | 0.498s |
| IGMN | 40 | 0.034598 | 1.0 | 0.351s |

We can notice in Table 1 that the IGMN performance is comparable to other ANN models. In fact, just the GRNN model has a better performance, but observing the boxplot graphs in Figure 4 we can notice that the differences are not statistically significant, i.e., the confidence intervals overlap. Moreover, IGMN can learn the target function very fast using a single scan over the training data and does not require an exhaustive search to find out the best configuration parameters (actually just the $\varepsilon_{max}$ parameter must be reduced until the desired approximation level is achieved).

The next experiment was performed in a more complex and irregular environment, shown in Figure 5, where the robot was preprogrammed to follow the external walls of the simulated environment. IGMN was trained using the data corresponding to one lap in the environment (1631 samples), and was tested using another independent lap (1551 samples). This experiment is more difficult than the previous one (Figure 2(b)) because the control algorithm is more complex. In fact, in the previous experiment the robot was manually controlled to perform a fixed trajectory in the regular environment, whilst in this experiment the robot is automatically controlled using the noisy sonar data to perform a wall following behavior, and therefore the robot's trajectory changes slightly at each lap according to the noisy readings received at each instant.

The solid gray curve in Figure 5 shows the trajectory followed by the robot during the learning phase, and the dashed black curve shows the trajectory followed by the robot using the trained IGMN network to control its actions. We can notice in this

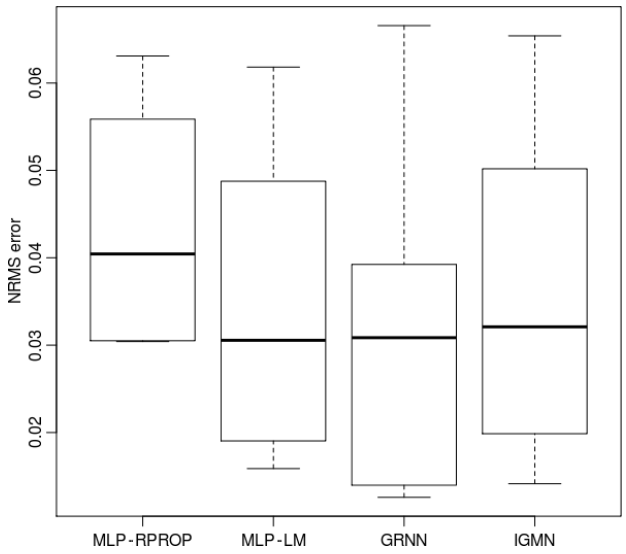



Figure 4: Comparing the approximation errors in the trajectory of Figure 2(b)

figure that the robot have not simply repeated the "target trajectory" after training. On the contrary, it follows a softer trajectory through the environment, which demonstrates that the neural network has really learned the wall-following behavior rather than just reproducing the target trajectory.

The configuration parameters used in this experiment are $\delta = 0.01$ and $\varepsilon_{max} = 0.75$, and just two Gaussian units were added during learning. Actually using lower values in $\varepsilon_{max}$ (e.g., $\varepsilon_{max} = 0.1$) the robot's trajectory will be more similar to the target one (and more neurons will be added to each region, of course), but from a practical point of view this is not interesting. As a matter of fact, it is much better learning a control behavior (a wall following behavior, in this case) than just reproducing a target trajectory, because the control behavior is much more robust against changes in the environment.

Table 2 shows a comparison among the results obtained in this experiment using IGMN and other ANN approaches. We can notice in this table that the IGMN performance is comparable to other models even using just two Gaussian units and without fine-tunning its configuration parameters. These experiments show that IGMN is a very suitable tool for robotic control tasks, because it can learn a control behavior very fast, incrementally and using few Gaussian units.

Table 2: Comparative among ANNs in learning the wall-following behavior

| **ANN model** | **Units** | **NRMS** | **Epochs** | **Time** |
|---|---|---|---|---|
| MLP – RPROP | 6 | 0.168697 | 78.2 | 4.224s |
| MLP – LM | 6 | 0.157162 | 27.9 | 7.146s |
| GRNN ($\sigma = 210$) | 1631 | 0.142121 | 1.0 | 0.229s |
| IGMN | 2 | 0.140875 | 1.0 | 0.045s |

## 4 Conclusion

This paper has presented the use of IGMN in on-line tasks such as incremental concept formation and robotics. IGMN is a new connectionist approach proposed at [2, 3] for incremental function approximation and on-line prediction. The main advantages of IGMN that makes it useful for these kind of task are:

- IGMN learns instantaneously using a single scan over the training data;
- It does not require that the complete training data set be available at the beginning of the learning process (each training pattern can be immediately used and discarded);
- It can create good estimates using few training data, and these estimates are improved as more training data arrive;
- The learning process can proceed perpetually;
- It handles the stability-plasticity dilemma and does not suffer from catastrophic interference;
- The network topology is defined automatically and incrementally (new units added whenever is necessary);
- It approximates the optimal Bayesian regression surface;
- It is not sensible to initialization conditions;
- It has few configuration parameters that are easy to set.

The performed experiments have shown that IGMN is a very useful machine learning tool for incremental tasks such as concept formation, robotic control and mapping tasks. As future work we plan to use IGMN with data provided by a real Pioneer 3-DX mobile robot.

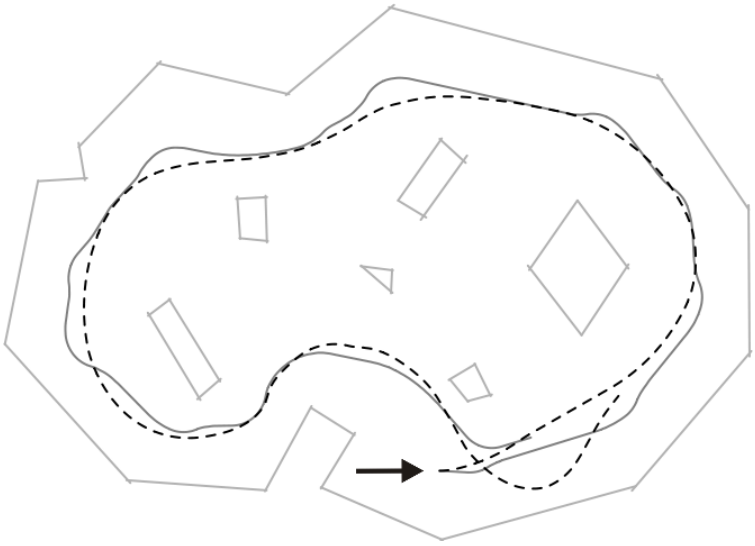


Figure 5: Wall following behavior in a more complex environment

## References

[1] S. Haykin. *Neural Networks and Learning Machines*. Prentice-Hall, Upper Saddle River, NJ, third edition, 2008.

[2] M. R. Heinen. "A Connectionist Approach for Incremental Function Approximation and On-line Tasks". Ph.D. thesis, Informatics Institute – Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, RS, Brazil, March 2011.

[3] M. R. Heinen and P. M. Engel. "IGMN: A Connectionist Approach for Incremental Function Approximation". *IEEE Trans. Neural Networks*, 2011. Under review.

[4] M. R. Heinen, P. M. Engel and R. C. Pinto. "IGMN: An Incremental Gaussian Mixture Network that Learns Instantaneously from Data Flows". In *Proc. of VIII Artificial Intelligence National Meeting (ENIA)*, Natal, RS, Brazil, July 2011. To appear.

[5] J. Hawkins. *On Intelligence*. Owl Books, New York, NY, 2005.

[6] P. M. Engel and M. R. Heinen. "Concept Formation using Incremental Gaussian Mixture Models". In *Proc. 15th Iberoamerican Congr. Pattern Recognition (CIARP)*, volume 6419 of *LNCS*, pp. 128–135, São Paulo, SP, Brazil, November 2010. Springer-Verlag.

[7] P. M. Engel and M. R. Heinen. "Incremental Learning of Multivariate Gaussian Mixture Models". In *Proc. 20th Brazilian Symposium on AI (SBIA): Advances in Artificial Intelligence*, volume 6404 of *LNCS*, pp. 82–91, São Bernardo do Campo, SP, Brazil, October 2010. Springer-Verlag.

[8] V. Strassen. "Gaussian Elimination is not Optimal". *Numerische Mathematik*, vol. 13, no. 3, pp. 354–356, 1969.

[9] D. Burfoot, M. Lungarella and Y. Kuniyoshi. "Toward a Theory of Embodied Statistical Learning". In *Proc. 10th Int. Conf. Simulation of Adaptive Behavior (SAB 2008)*, volume 5040 of *LNCS*, pp. 270–279, Berlin, Heidelberg, 2008. Springer-Verlag.

[10] S. Thrun, W. Burgard and D. Fox. *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents. The MIT Press, Cambridge, MA, 2006.

[11] J. H. Gennari, P. Langley and D. H. Fisher. "Models of Incremental Concept Formation". *Artificial Intelligence*, vol. 40, no. 1-3, pp. 11–61, 1989.

[12] D. Filliat and J.-A. Meyer. "Map-Based Naviagation in Mobile Robots: I. A review of localization strategies". *Cognitive Systems Research*, vol. 4, no. 4, pp. 243–282, 2003.

[13] S. Nolfi and J. Tani. "Extracting Regularities in Space and Time Through a Cascade of Prediction Networks: The Case of a Mobile Robot Navigating in a Structured Environment". *Connection Science*, vol. 11, no. 2, pp. 125–148, 1999.

[14] F. Linåker and L. Niklasson. "Time Series Segmentation Using an Adaptive Resource Allocating Vector Quantization Network Based on Change Detection". In *Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Networks (IJCNN 2000)*, pp. 323–328, Los Alamitos, CA, 2000.

[15] F. Linåker and L. Niklasson. "Sensory Flow Segmentation Using a Resource Allocating Vector Quantizer". In *Proc. Joint IAPR Int. Workshops on Advances in Pattern Recognition*, pp. 853–862, London, UK, 2000. Springer-Verlag.